RU Staying

Software Engineering
01:332:452
Report 2: Part 1

By Group #11
Keya Patel
Zain Sayed
Mohammed Sapin
Purna Haque
Nga Man (Mandy) Cheng
Rameen Masood
Shilp Shah
Mathew Varghese
Thomas Tran
Eric Zhang

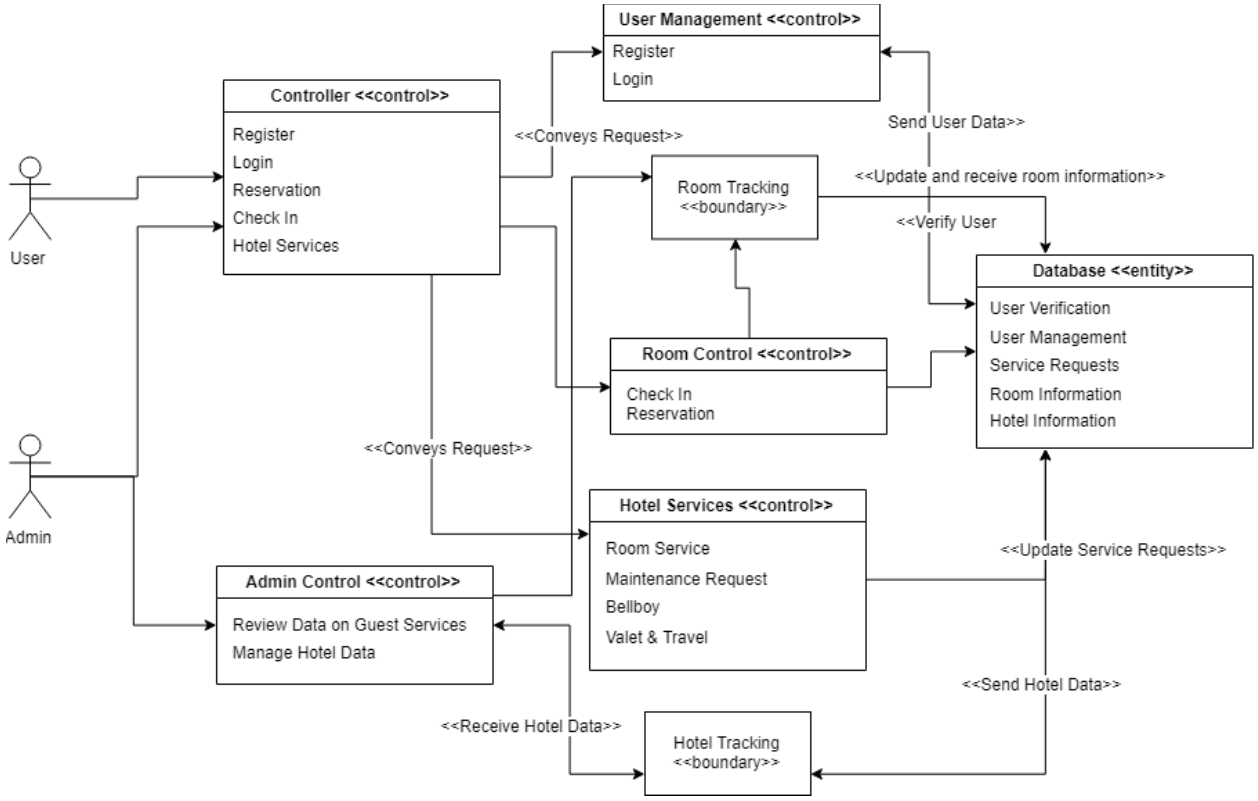
Github: https://github.com/mohammedsapin/RUStaying
Submission Date: March 3, 2019

Domain Model

Concept Definition and Responsibilities

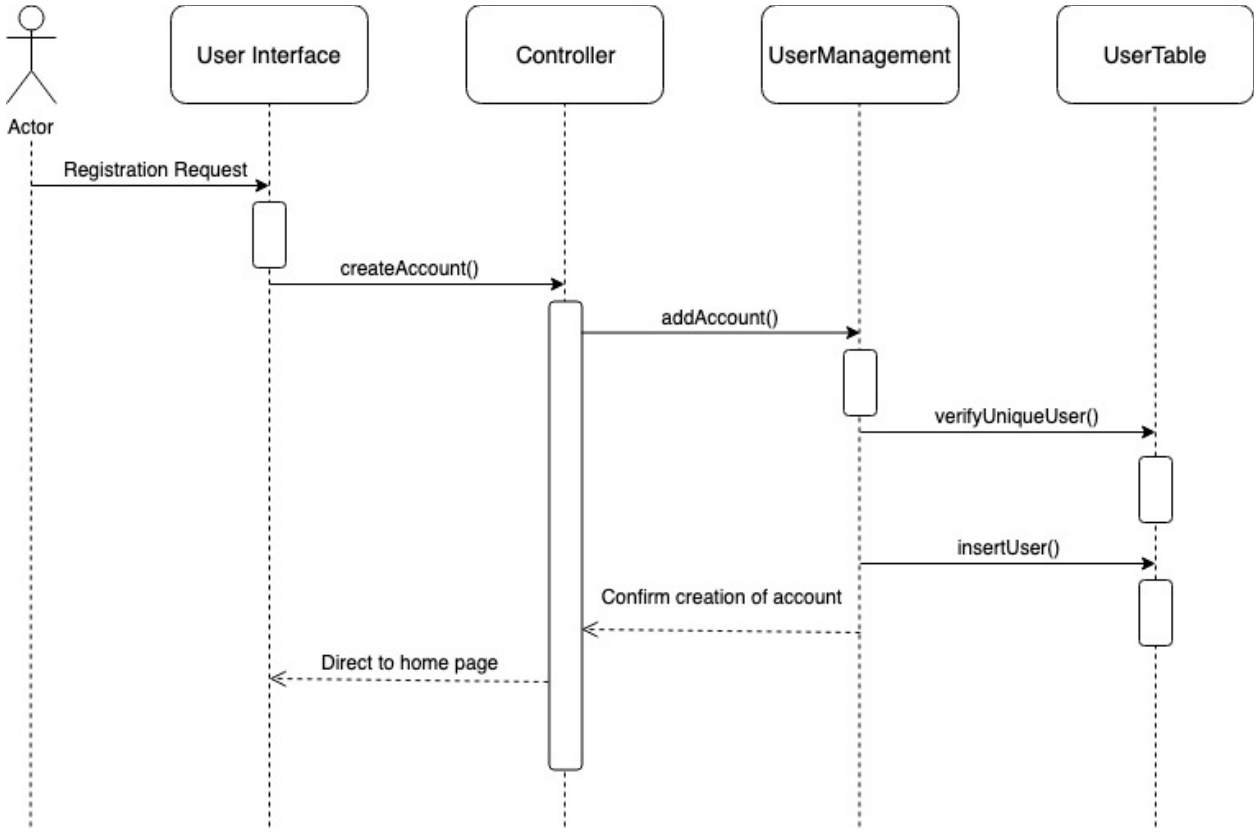
Responsibility Description	**Type**	**Concept Name**
RS1. Main source for all other subsystems to interact with. Coordinate actions and delegate work based on user interactions.	D	Controller
RS2. Verify if user credentials are valid	D	UserVerification
RS3. Create a new guest account	K	UserManagement
RS5. Store user account information (username, password, email, previous reservations etc…)	D	UserManagement
RS4. To allow user to make a reservation	D	RoomControl
RS6. To allow the guest to check-in for their hotel reservation	K	RoomControl
RS7. Guest can request room service	D	HotelServices
RS8. Guest can make maintenance requests for their room during their stay	D	HotelServices
RS9. Guest can request a bellboy for luggage	K	HotelServices
RS10. Guest can use the hotel car service	K	HotelServices
RS11. System will keep track of available / unavailable rooms	D	RoomTracking
RS12. System will keep track of each hotel service as it is requested	D	HotelTracking
RS13. Review all data on services used by guests	D	AdminControl
RS14. Present data in viewable manner to analysis and predictions	D	AdminControl

Domain Model Diagram



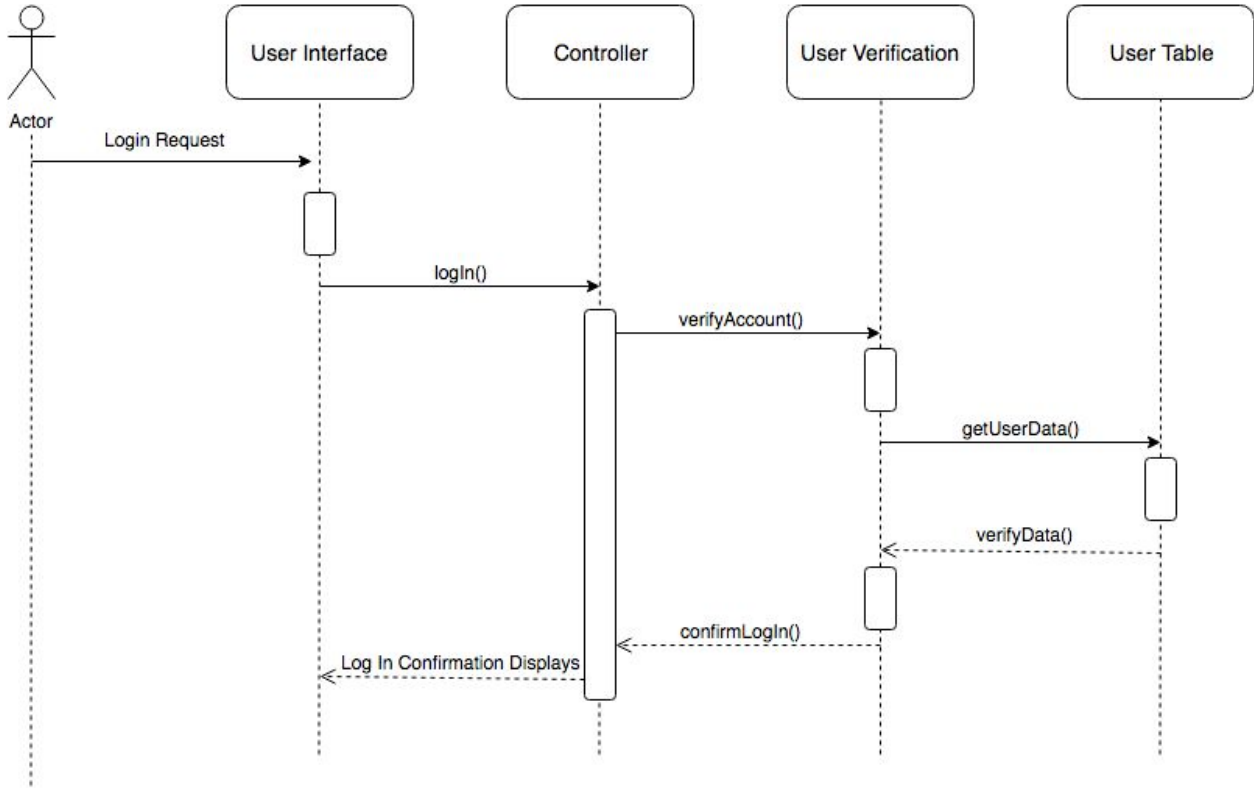
System Sequence Diagrams

Use Case 1 - Registration



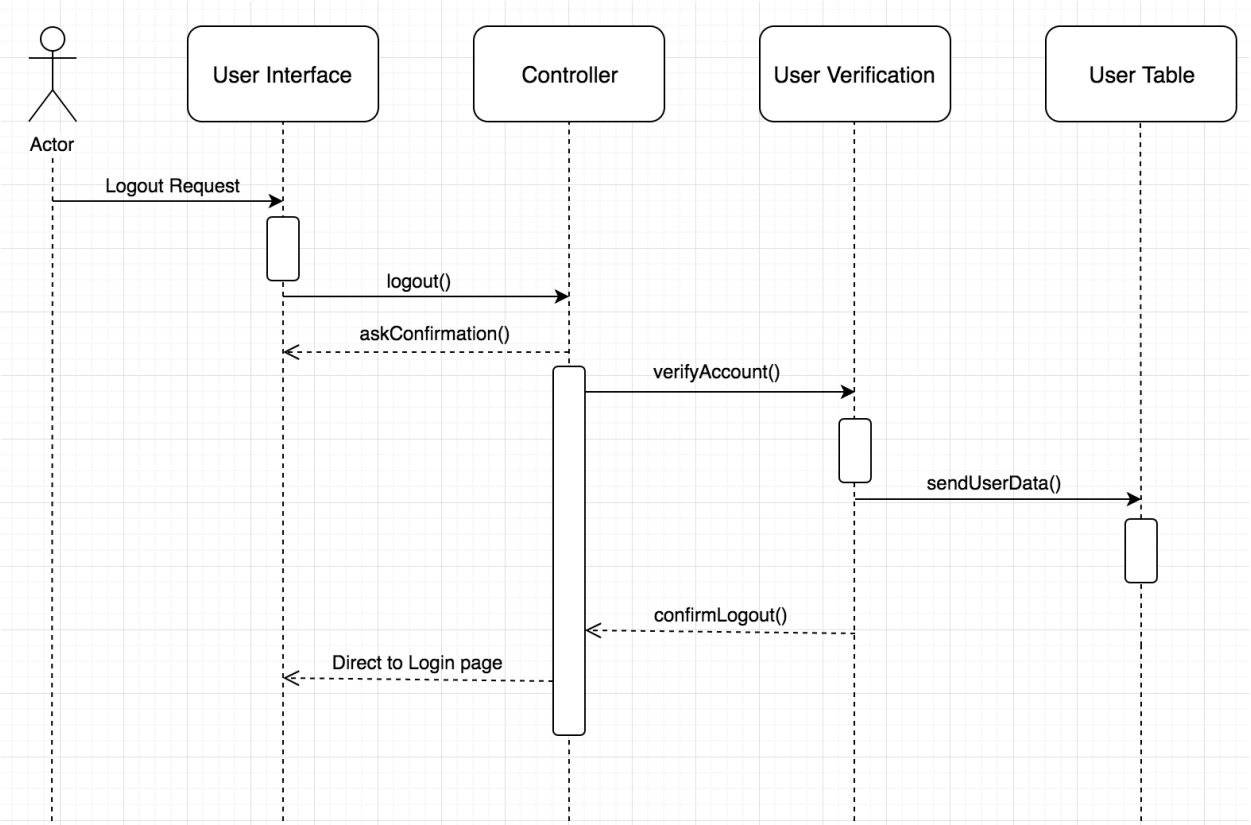
In this use case, the user can register for a new account. This is a relatively straight forward process. Once the user fills out the required information for a new account, the main controller will send the information to the UserManagement controller. The UserManagement controller is responsible for keeping track of all current and new accounts with the app RUStaying. It directly interacts with the database, specifically the UserTable, which is a table with all current accounts and their corresponding information. The UserTable concept was not listed in the domain model because we grouped all the specific database related entities. However, in these interaction diagrams, it is important to separate the different tables and information the database will be storing. The UserManagement controller will verify with the UserTable if this is a unique account and then process to insert the new user to the table. Once the main controller gets a confirmation that the user was created, it redirects the User Interface to the main home page so the user can access the hotel services through the app.

Use Case 2 - Login



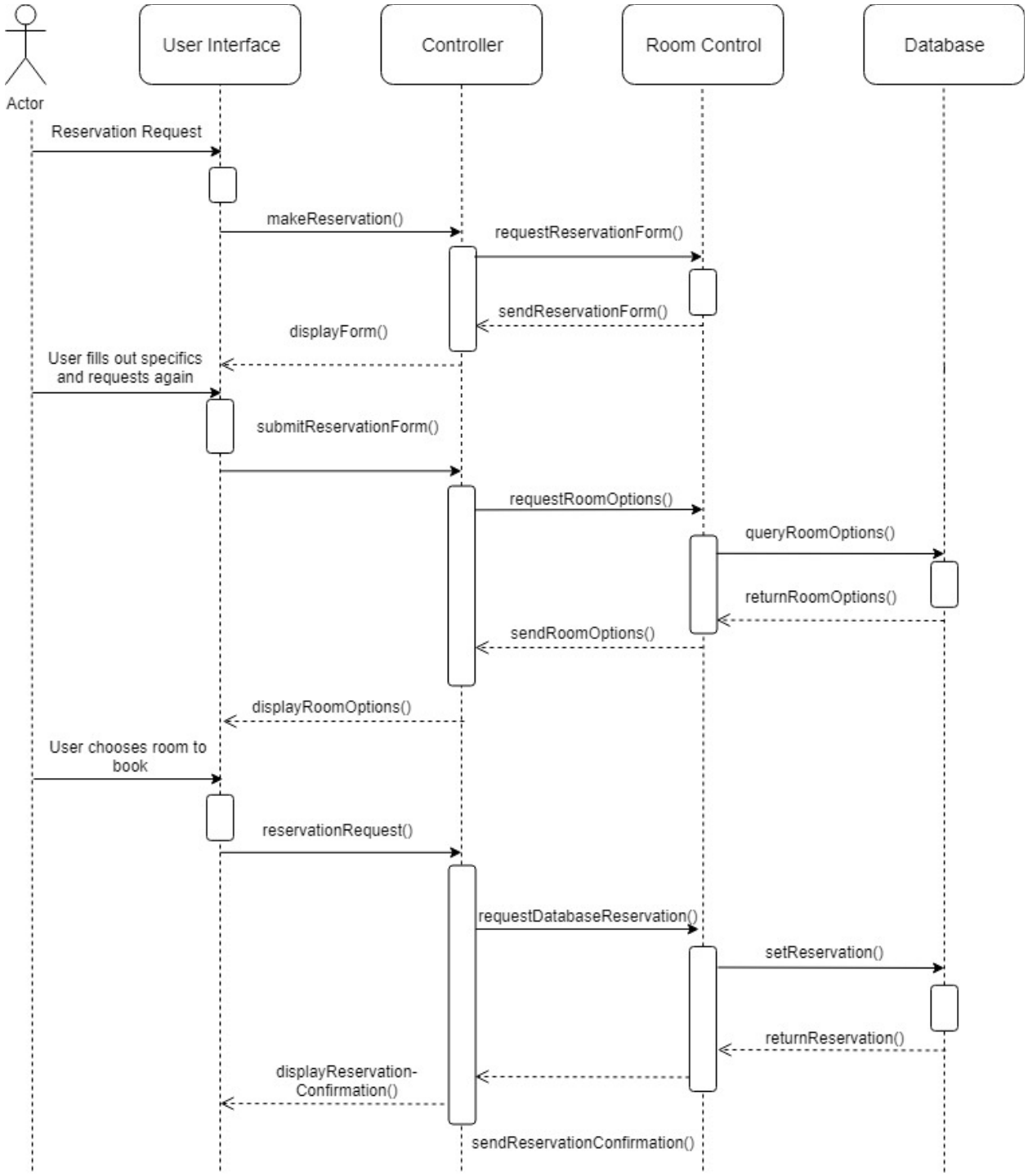
In this use case, the user can log into their account. Once the user inputs their username and password into the interface, the main controller will send the information into the UserVerification control. This is responsible for sending the data submitted by the user to the UserTable and verify if the information entered corresponds with an account that is in the database. If the account information is confirmed, the UserTable will send a verification to the UserVerification if the account information matches that of an existing account. The UserVerification will then confirm the login and allow the user to log in. If the information entered by the user does not correspond, then the user will be asked to input the login information again.

Use Case 3 - Logout



In this user case, the user is able to log out of an existing account. Once the user creates a logout request, the main controller will send a confirmation message back to the user to confirm logout. Then, the controller will send the information UserVerification control. User Verification will send the data submitted by the user to the User Table. The UserVerification will then confirm the logout and allow the user to logout of their account. The user will be redirected to the login page.

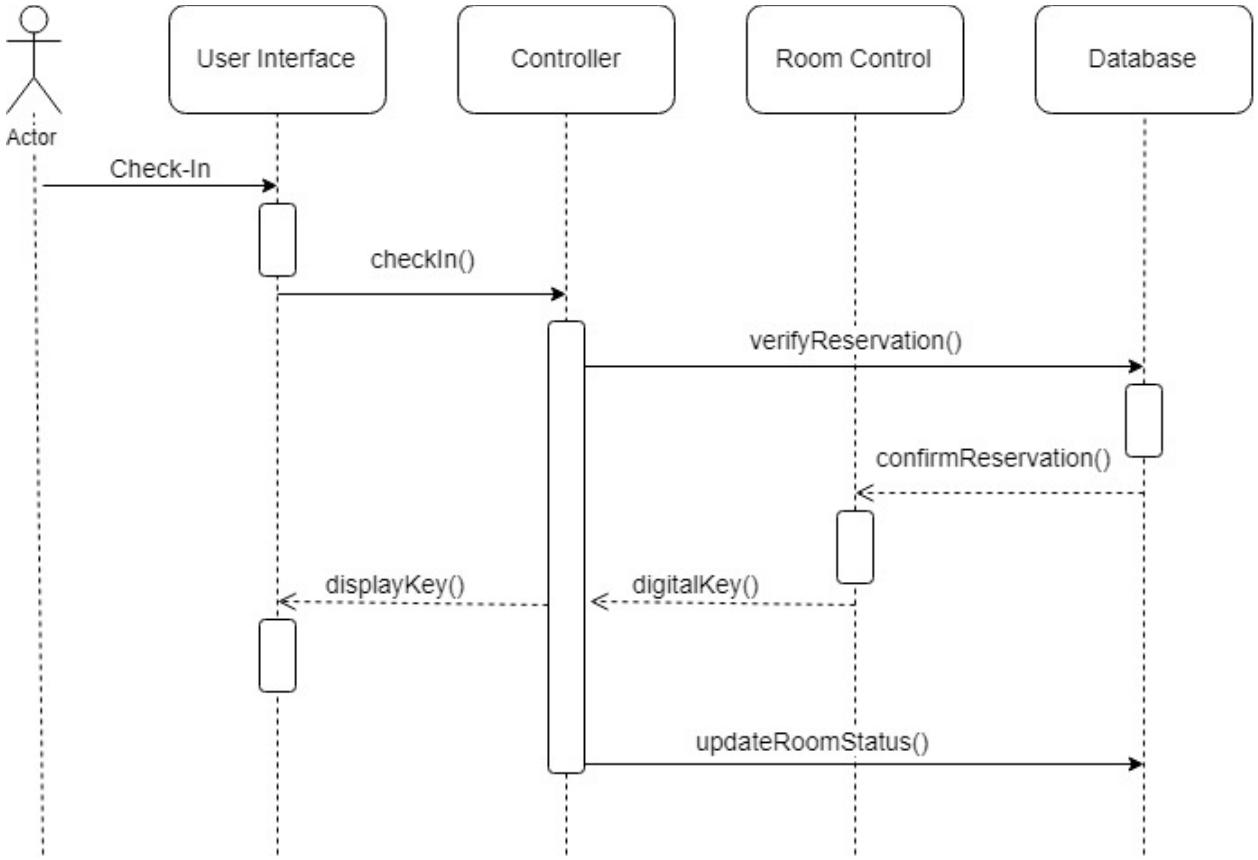
Use Case 4 - Make a Reservation



In this use case, the user will be able to make/book a reservation. This process can be roughly broken down into 3 parts. The first part, user will request to make a reservation, prompting the controller to forward the request to Room control. Room control will then send a form to user through controller for filling out. Room control does not need to access database therefore

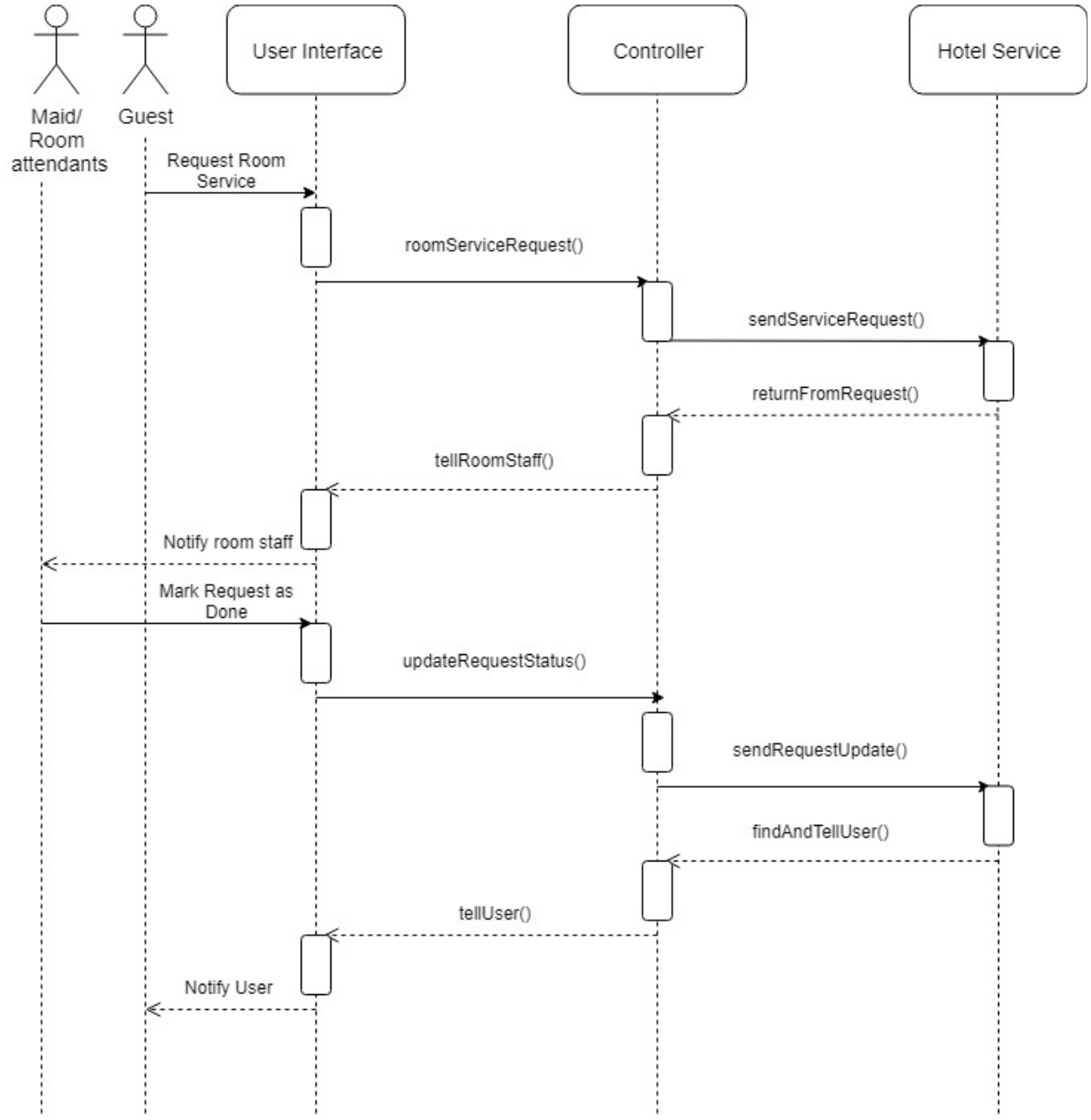
cutting down on over head for this part. Once form is filled out, controller will take form and address the input for room control, so room control can now handle retrieving rooms similar to those requested by guest. Room control will then send that info back to controller so it may display it for user to pick and choose. Once chosen, user will initiate final stage wherein, controller will tell room control to book the request and finalize it through database(Database needs to show that room is booked for however long guest is staying).On fulfillment, database will return confirmation through controller.

Use Case 5 - Check In



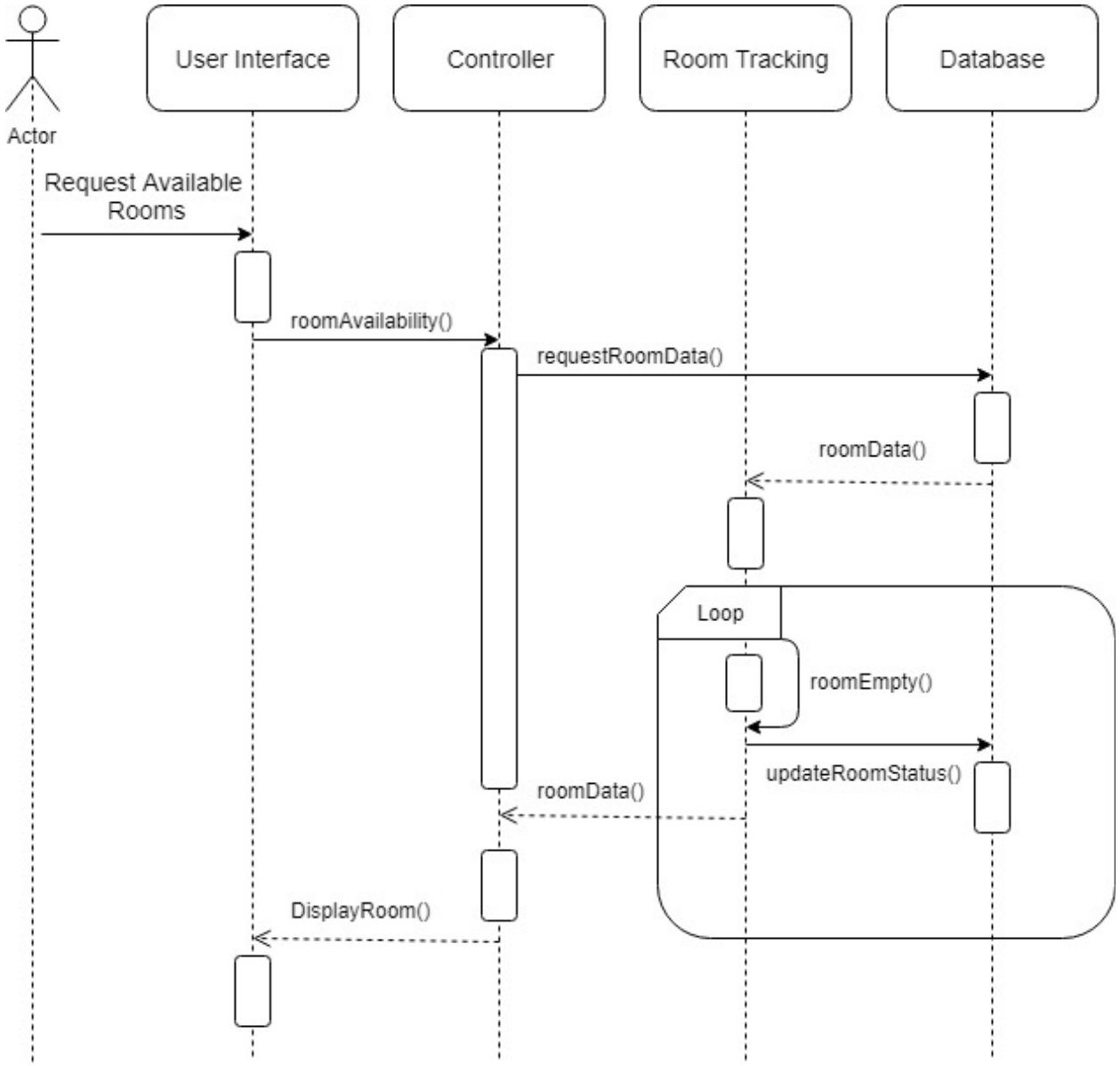
UC-5 refers to how a user can check into their hotel room using our application. It starts off with us assuming user has logged in, clicks check-in button on the user interface. The controller will run the check in function. From here we need to make sure the user has a reservation so the controller will make a request to the database to verify the reservation. The database will confirm the reservation and send it to the room controller so they can provide the user a key for the room and check the user in. After that the control sends the request to display the key on the user interface as well as update the room status in the database.

Use Case 6 - Request Room Service



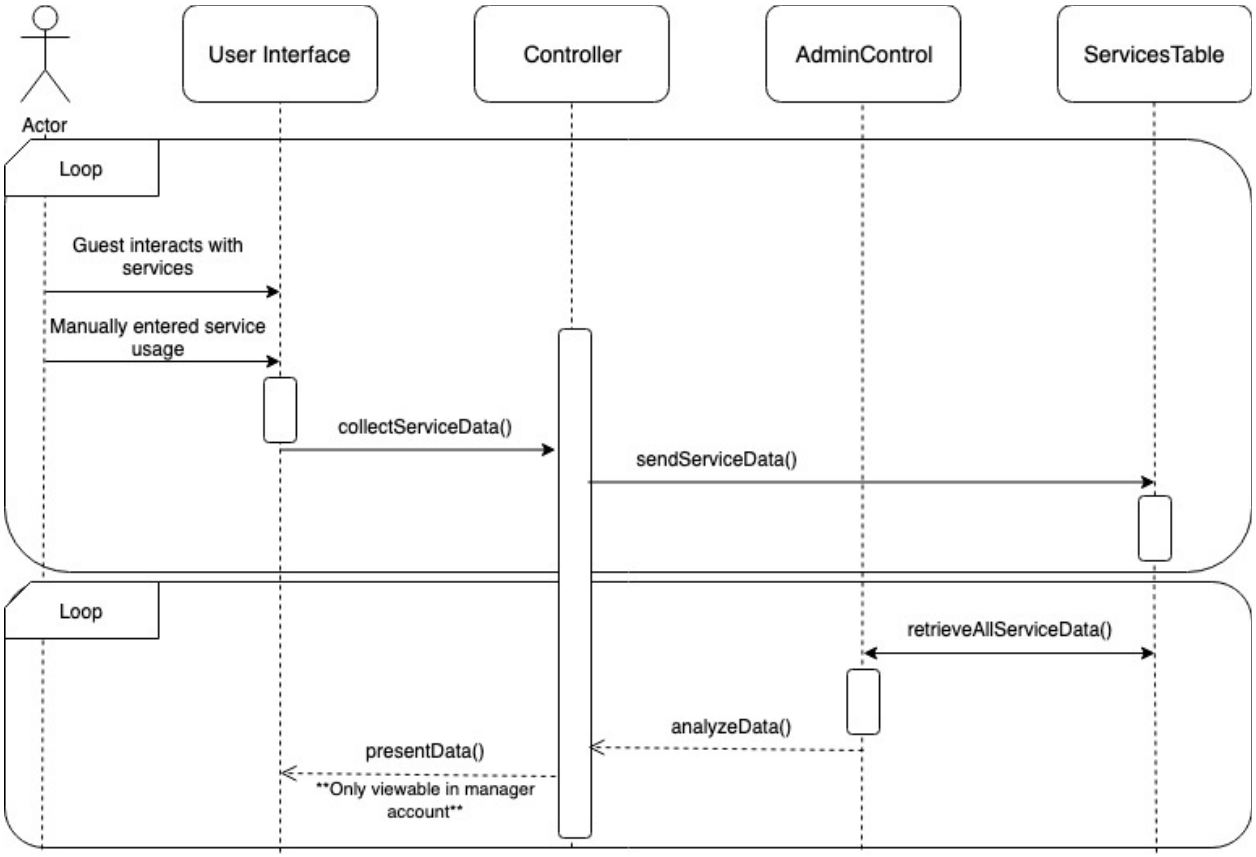
In this use case, the user will be able to call for room service(cleaning, replacing sheets, etc).One of the many functionalities of the app is streamlined requests like this. To access this service, user will input a request from the UI through which controller will receive a signal to Hotel Services which handles these calls. Hotel Services will then forward the request to room attendants to fulfill said request and once request is done, request to mark as done will be passed through controller back to Hotel Services. Once it has been marked as finished, guest will receive a notification of completion as well.

Use Case 7 - Check Availability of Rooms



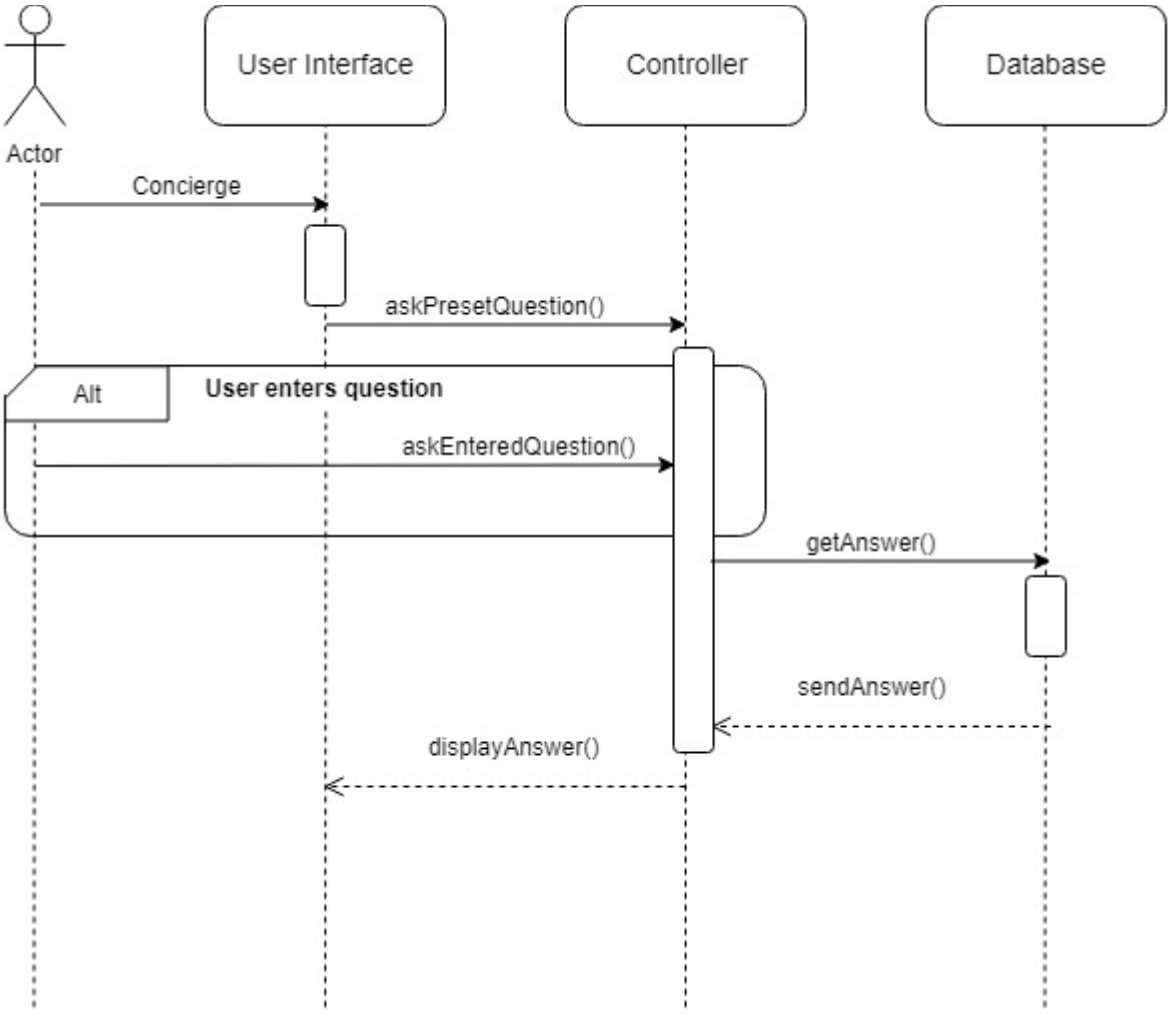
UC-7 allows a user to check the availability of rooms at the hotel. Once they make the request the controller request room data from the database. The database sends this information to the room tracking boundary. Room tracking will run a loop to check if the room is vacant or not. Once the loop runs it will update the status of the room in the database as well as send the data to the controller. The controller then sends the request to display the information on the user interface on whether a room is vacant or not.

Use Case 8 - Review Guest Data



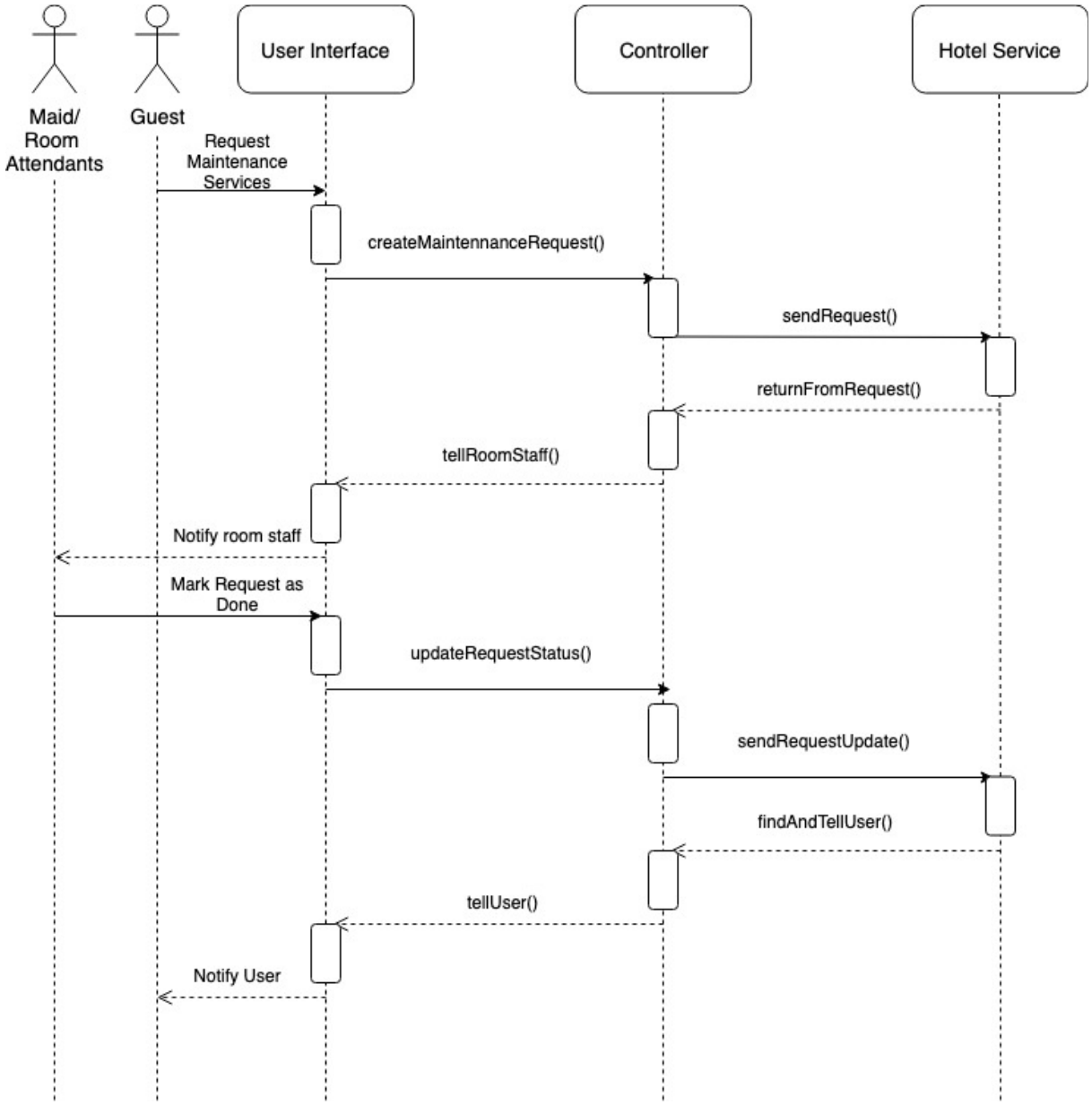
One of the benefits of having an app for hotel automation, is the ability to collect and analyze data. This is a very important use case because it allows administrators to see the usage of the different services provided by the hotel. As guests use the services, or as staff members manually enter usage from guests who do not use the app, the data will be collected by the main controller and sent to the database for storage. Specifically, within the database, we will have a ServicesTable which stores this data along with the frequency and time it was used. The ServicesTable was not in the concepts from the first report because we grouped it together under Database. This process is encapsulated in a loop because as guests continue to use services, the data will be sent and updated in the ServicesTable. The second part of the process involves the AdminControl concept to collect the data from the table. Within AdminControl, we will implement simple data analysis algorithms to be able to present the data back to the administrator. This is also in a loop because as the ServicesTable is updated, the AdminControl must also update the statistics presented.

Use Case 9 - Concierge



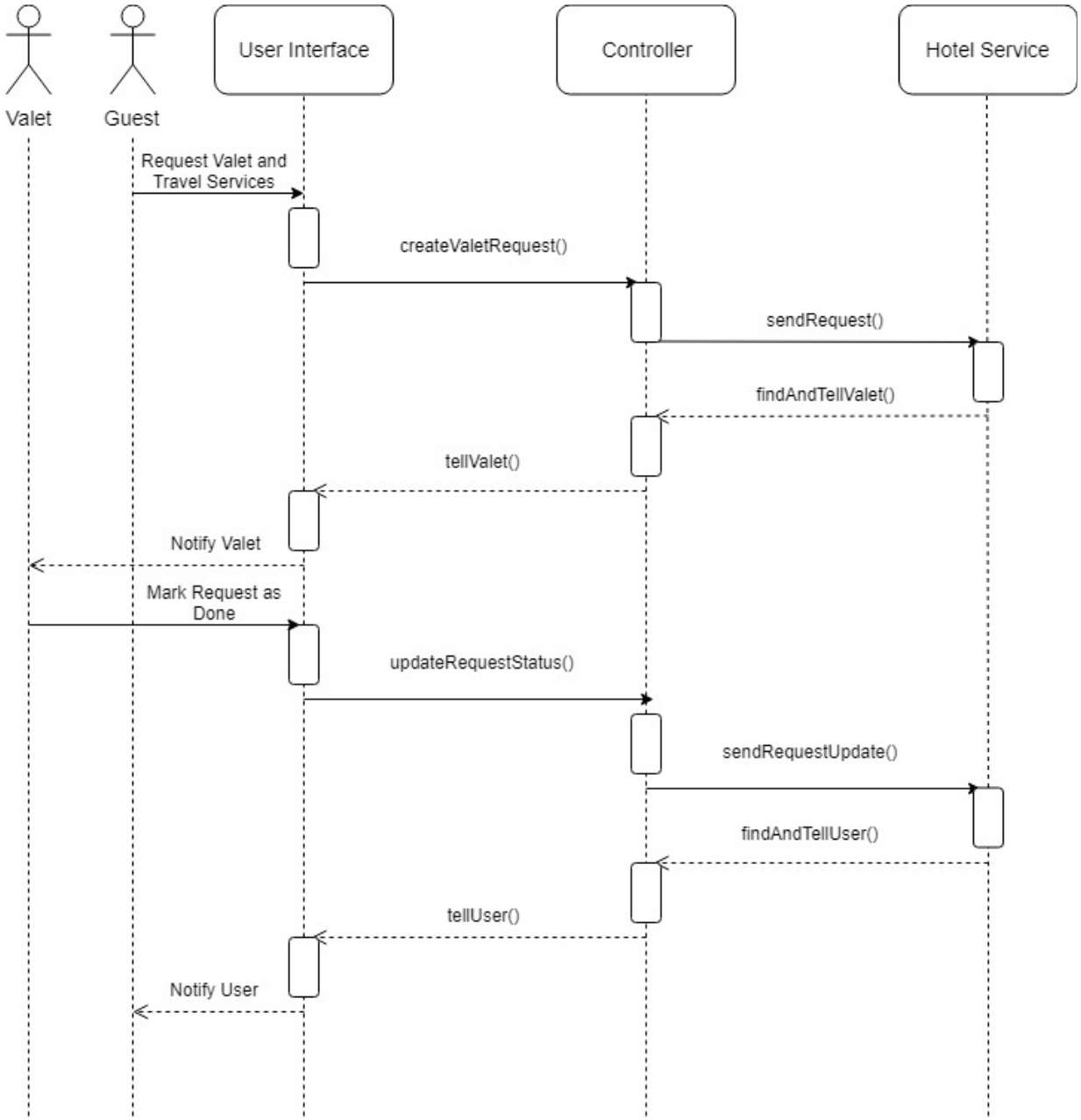
The concierge tab will allow users to utilize an automated concierge service, providing a quick way for users to ask questions without needing to leave their rooms. The process is as follows: the user will first navigate to the concierge tab in the app, then be shown a list of preset frequently asked questions to choose from. Or, if the user cannot find their question in the preset list, they can enter their own question. The question contained in the askPresetQuestion() and askEnteredQuestion() functions are sent to the database, where getAnswer() will interpret the question and search for an answer. Using sendAnswer(), the database returns the information back to the application for the controller to display to the user using displayAnswer(). If getAnswer() cannot find an answer in the database, sendAnswer() will automatically return a message displaying the front desk concierge hours and phone number, directing the user to call or see the front desk later for help.

Use Case 10 - Maintenance Requests



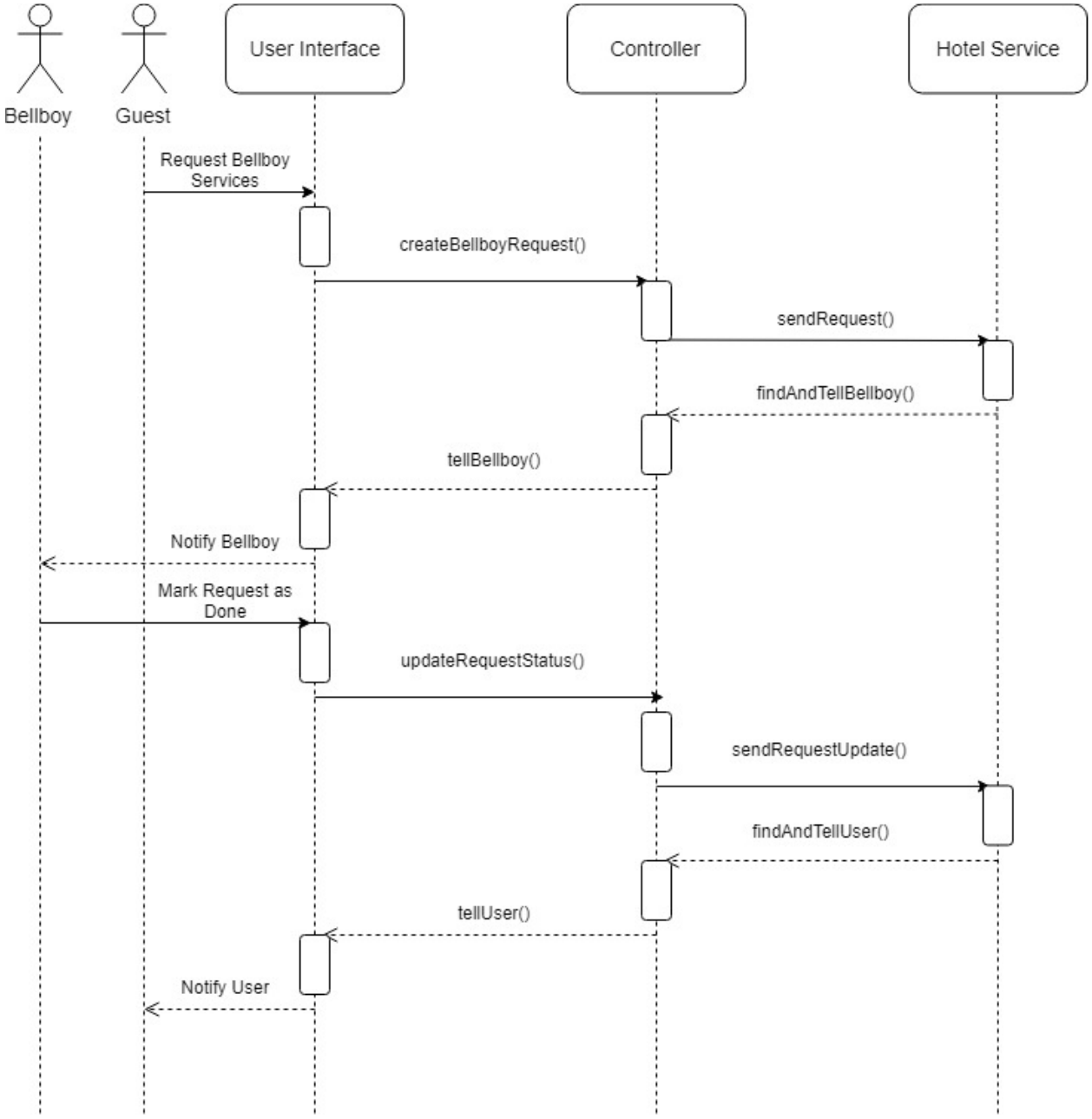
The maintenance requests tab will allow for an efficient way for guests to access request forms for problems within their rooms. Through the app, they will directly be able to navigate to the maintenance request tab, then guests will be able to directly submit their complaint. This is part of the createMaintenanceRequest() function. Consequently, the request will be sent to the database (sendRequest()) and returned to the room staff (tellRoomStaff()) which will allow the room staff to be notified. Once the request has been handled by the room staff and update of the status will be sent through the updateRequestStatus() and then it will be sent to the database. After this, the guest will be informed about the update through the findAndTellUser() and tellUser() functions.

Use Case 11 - Valet & Travel Services



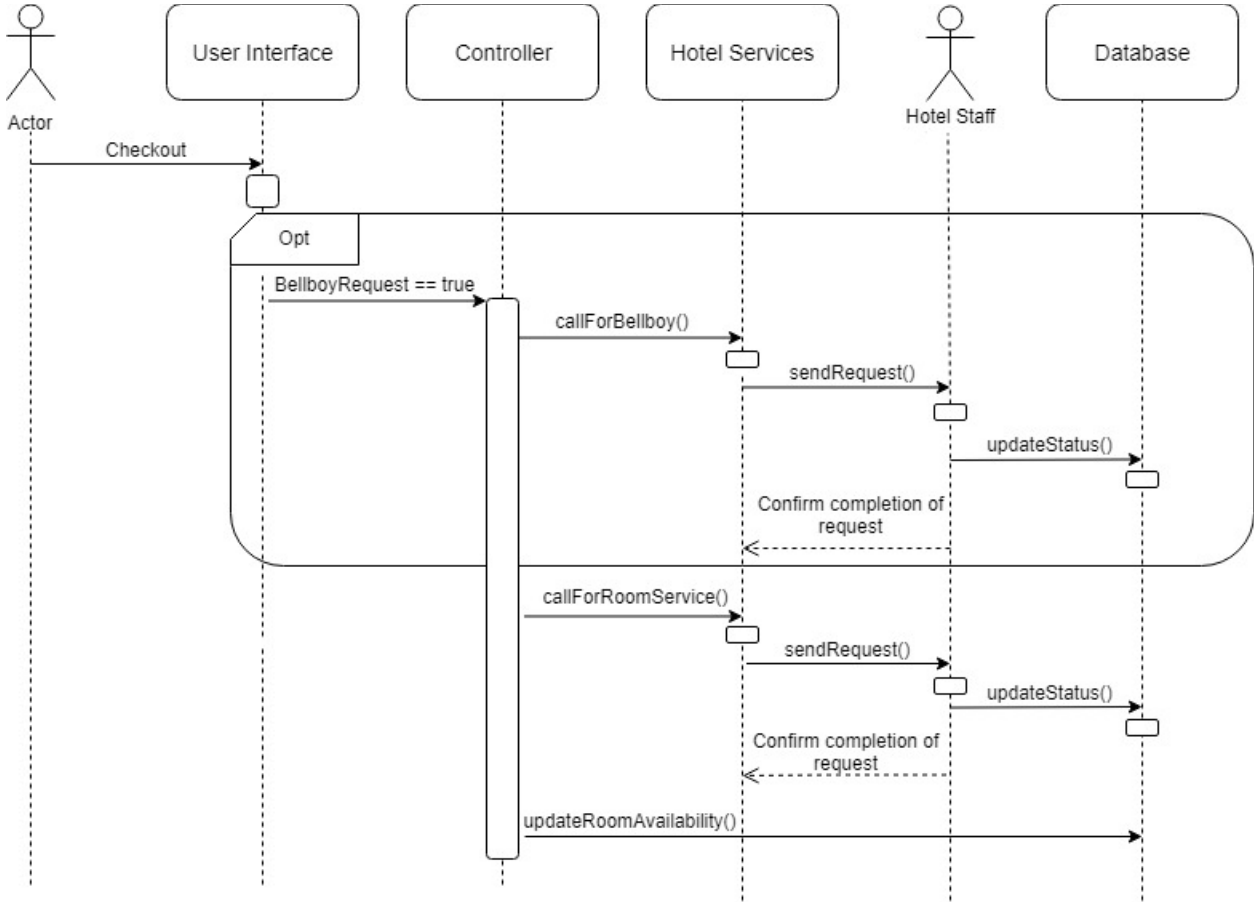
One of the many hotel services includes valet and travel services. This use case is a fundamental service within a hotel. As the guest sends a valet or travel request, the user will specify if they will need to be transported or to have parking services as they submit a request. As staff members, they will receive the guests requests and work to complete the quests requests. They will use the app from the administration side and update the status of the request in the database. The user will receive a confirmation of the complete requested and be notified and will be directed to the home page. This is an outline of how valet and travel services will be processed.

Use Case 12 - Bellboy Requests



A bellboy system was decided upon to make sure that customers could easily get their luggage taken to their room. The guest will tap on their phones to make a request. This request will be sent to the Bellboy in the form of a notification on his/her app. The bellboy can mark this request as done, and this information will eventually be sent to the guest who will receive a notification that their luggage has been moved to their rooms.

Use Case 13 - Checkout



The checkout option through the app is essential to confirm the user has finished their stay at the hotel. After the user has chosen to checkout, they are given the option to request bellboy service to help bring their luggages down to the lobby. On the other hand, room service is automatically notified that they are needed in the room the user has just checked out of. Staff members, bellboys and maids in this case, will use the app to receive their respective requests and to update the statuses of them as well. As soon as the maids have updated the status of this specific request to completed, this will automatically trigger the database to update the number of available rooms.

Project Management

Merging Contributions

While working on Report 1, we realized it is very important to maintain consistency between all individual members. Because the work is split between multiple groups of people, there were variations in our diagram layout and formatting. We worked to fix this issue by the time the full Report 1 was due, but now starting with Report 2 we are more aware to address this early on. All team members will now be using draw.io for the diagrams and we all are following an uniformized format for the report. This will allow us to put together each members contributions easily into one report without much variation.

Project Contribution and Progress Report

So far, we have implemented the initial functionality of the app. This took a little longer than originally expected because most of our group members were learning how to use Android Studio. Once we were all set up, we started designing the different XML pages and working to transition between them. Next, we set up the database functionality with Android Studio so we can store and retrieve information. Setting up SQLite was a little challenging because no one in our group had worked with it before. Now all the parts of the app were ready for use and we started implementing the functionality behind the user interface. As of now, the guest is able to register for a new account, login to existing accounts and logout of their account. All of this is handled properly, while checking for all possible error conditions as well. This corresponds to the completing Use Cases 1, 2 and 3. We had a couple group members implement this functionality but a lot of other team members also contributed to become familiar with Android Studio. This will definitely not be the situation with the other Use Cases and we will need to use Github to our benefit to branch and merge code efficiently.

Plan Of Work

Since the first report, our plan of work has slightly adjusted and we have modified our milestones to have a more in depth analysis of the requirements and tasks that need to be completed.

Milestones

1. Finalized front end design for login pages
    a. This is the very beginning of our app and these pages we ones most commonly used. Based on the mock-ups from Report 1, we started to recreate these pages in Android Studio. These pages are necessary before we can even begin any functionality of the app. The specific pages this milestone references are:
        i. Login Page
        ii. Registration Page
        iii. Home Page (after user logs in)
    b. Status: Completed
2. Design all other pages the user can navigate to in the app
    a. Look at the Navigation Tree created in report 1 and start to create all those pages in Android Studio. Also, maintain consistency with the current pages already created and follow same color theme. This is just an user interface to be able to implement functionality with later on.
    b. Status: In Progress
    c. Due: 3/8/19
3. Allow guests to login to existing accounts
    a. Check with the database table if the user credentials (email and password) are correct. Redirect guest to home page if login successful or notify guest of invalid login credentials
    b. Status: Completed
4. Administrator Account Login
    a. Create preset email addresses and passwords that indicate a manager account. The manager account will have a slightly different user interface but most pages should be reused from the guest user interface. This milestone is to design the new XML pages specifically for an admin and to implement the functionality for an admin to login
    b. Status: Not Started
    c. Due: 3/8/19
5. Check availability of rooms
    a. Create a database table with all the types of rooms available at the hotel, each with an available/unavailable flag. In the table, we will also have columns for the

duration the specific room is being used for. The table is collection of all the rooms and the exact dates they are reserved for. This functionality is necessary for other use cases and keeping track of data.

   b. Status: Not Started
   c. Due: 3/8/19

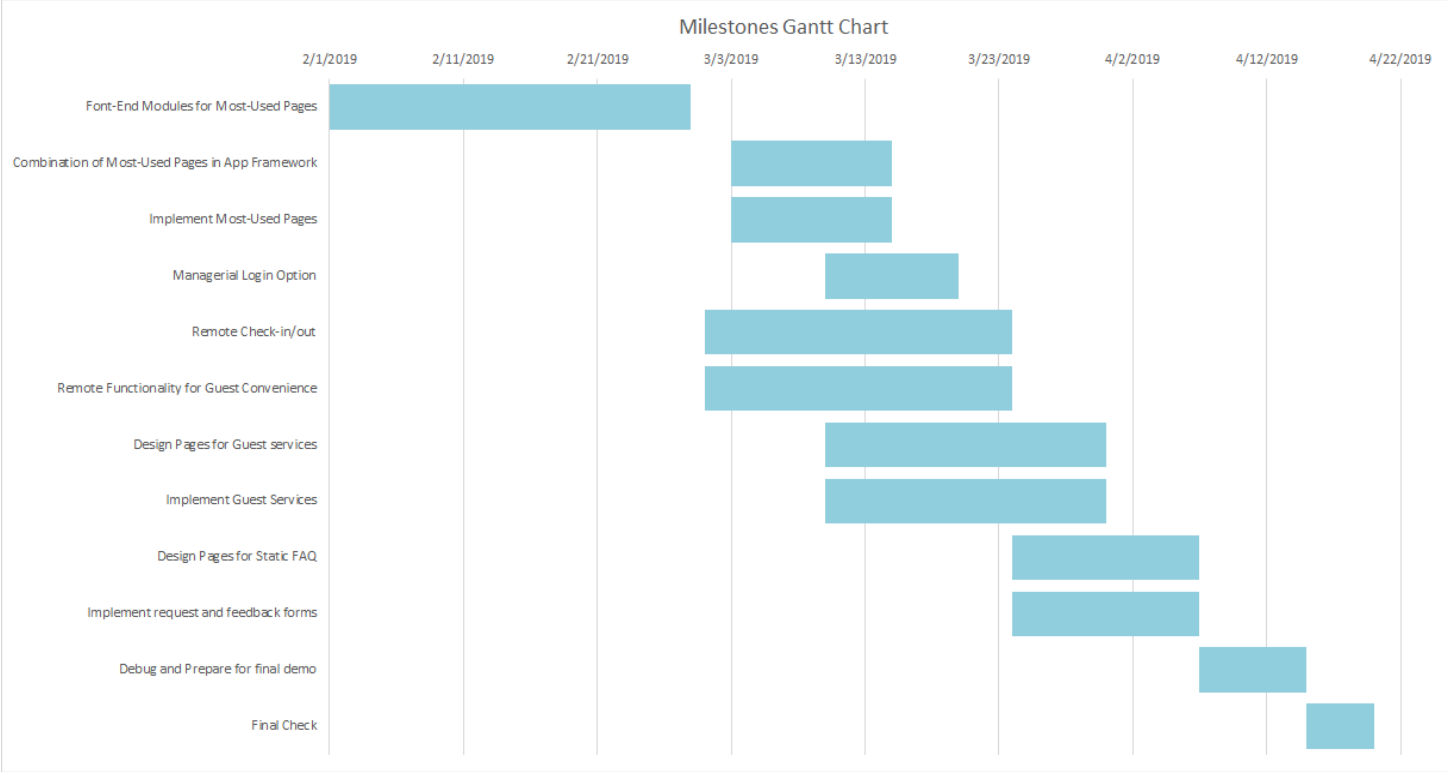
6. Implement Make a Reservation functionality
   a. One of the most used features, is to make a reservation at the hotel. This corresponds to Use Case 4. To be able to implement this use case, we will need to use the functionality of the previous milestone. The user will be prompted to enter information about the room and the duration of their stay with exact dates. When a new reservation request comes in, we will check the table of all rooms and only present the guest with the rooms that have the "available" flag. This table will be constantly updated as guests make reservations, so that no unavailable rooms are accidently shown to guests. When a reservation is confirmed, the information will be stored in the UserTable under the appropriate guest. This milestone requires a few steps and is expected to be a little challenging so we allocated more time for it.
   b. **Status: Not Started**
   c. **Due: 3/15/19**
7. Allow guest to check-in for their stay
   a. This milestone is also dependent on the previous one, as a guest needs to have a reservation in order to check-in. This is a relatively simple milestone, as we only have to check if the guest has a reservation for a room. If the guest successfully checks into a room, then the room is going to marked with a flag to indicate that it is now in use.
   b. **Status: Not Started**
   c. **Due: 3/17/19**
8. Feedback form
   a. This case occurs when the guest has ended their stay at the hotel. We know the guest's stay has ended because when we made a reservation, they specified a start and end date. So the feedback form will be prompted when it's the guests last day at the hotel. The main part of this milestone is to design the form and prompt the guest on the last day, so it is relatively simple, but dependent on completing other milestones
   b. Status: Not Started
   c. Due: 3/17/19
9. Frequently Asked Questions Interface
   a. This will be a static page with common questions about using the app and hotel services. The purpose of this page is to gives the user instructions on how the app

works and some conveniences provided. This does not have much functionality so it should be relatively quick to implement.

   b. **Status: Not Started**
   c. **Due: 3/8/19**

10. Valet & Travel/ Bellboy Interface
    a. These two services are relatively similar in their functionality. The guest will request this service through the app and the staff will be notified of a new request made. The user will get a notification when the request is completed. We need to implement an interface that allows staff to give updates that will notify the guests of the status of their requests.
    b. **Status: Not Started**
    c. **Due: 3/17/19**
11. Debug and test implemented features
    a. The milestones listed above all due before the first demo (3/25). By meeting these due dates, we will have enough time afterwards to test our these features and fix any bugs. The purpose of this is to have a seamless software with enough functionality by the time of the first demo.
    b. **Status: In Progress**
    c. **Due: 3/24/19**

Gantt Chart


Milestones Gantt Chart
2/1/2019
2/11/2019
2/21/2019
3/3/2019
3/13/2019
3/23/2019
4/2/2019
4/12/2019
4/22/2019
Font-End Modules for Most-Used Pages
Combination of Most-Used Pages in App Framework
Implement Most-Used Pages
Managerial Login Option
Remote Check-in/out
Remote Functionality for Guest Convenience
Design Pages for Guest services
Implement Guest Services
Design Pages for Static FAQ
Implement request and feedback forms
Debug and Prepare for final demo
Final Check

Breakdown of Responsibilities

Keya Patel

- Set up user interface on administrator side that allows service data to be easily viewable
- Work with Thomas to see how data is sorted and provide accurate statistics

Zain Sayed

- Implement Check-In module and the corresponding pages needed
- Design feature for guest to choose start and end date of their stay with a calendar view

Mohammed Sapin

- Implement Make a Reservation module and design necessary pages
- Work together with Zain to setup the back-end database table of rooms available

Purna Haque

- Work on car/valet service alongside with room service. Create the necessary pages and the proper interaction with database
- Figure out interaction with Hotel Staff actor and how guest is notified of completed service

Nga Man (Mandy) Cheng

- Implement Bellboy service feature (work with Purna because the services work similarly)
- Allow guest to get a digital room key after they successfully check-in

Rameen Masood

- Implement Maintenance Requests module and design necessary pages
- Login and Registration modules. Also design and format additional XML pages

Shilp Shah

- Work on administrator side to collect service usage data and also design the respective unique pages
- Implement feedback form feature, which prompts the guest on the last day of their stay

Mathew Varghese

- Setup and maintain database of available and unavailable rooms.
- Get reservation and check-in information from other modules and update the table appropriately

- Maintain list of available rooms at all times to present the user when a new reservation request is being made

Thomas Tran

- Collate all the data stored by other modules in the database
- Implement simple data sorting and analysis algorithms to notice some trends in data

Eric Zhang

- Setup the static pages for Frequently Asked Questions
- Focus on proper integration between each team member
- Maintain consistency in user interface and object oriented design between group members

Integration

Eric Zhang will be coordinating the integration of all the different modules from group members. This responsibility is very important as it will keep our software code consistent and working seamlessly once integrated. He will work with each member to ensure that proper objects and tables are being used in each situation.

Integration Testing

Each unit will be thoroughly tested by each individual member before it can be integrated into the master branch. Shilp and Mohammed will perform the integration testing once modules are slowly added to the branch. Our main responsibility is to ensure that the different pages and functionality all work together with the main app. While testing, we will report bugs back to the original owner of the code to fix and provide an updated version. Most of this testing will be done prior to the demo to ensure smooth performance.